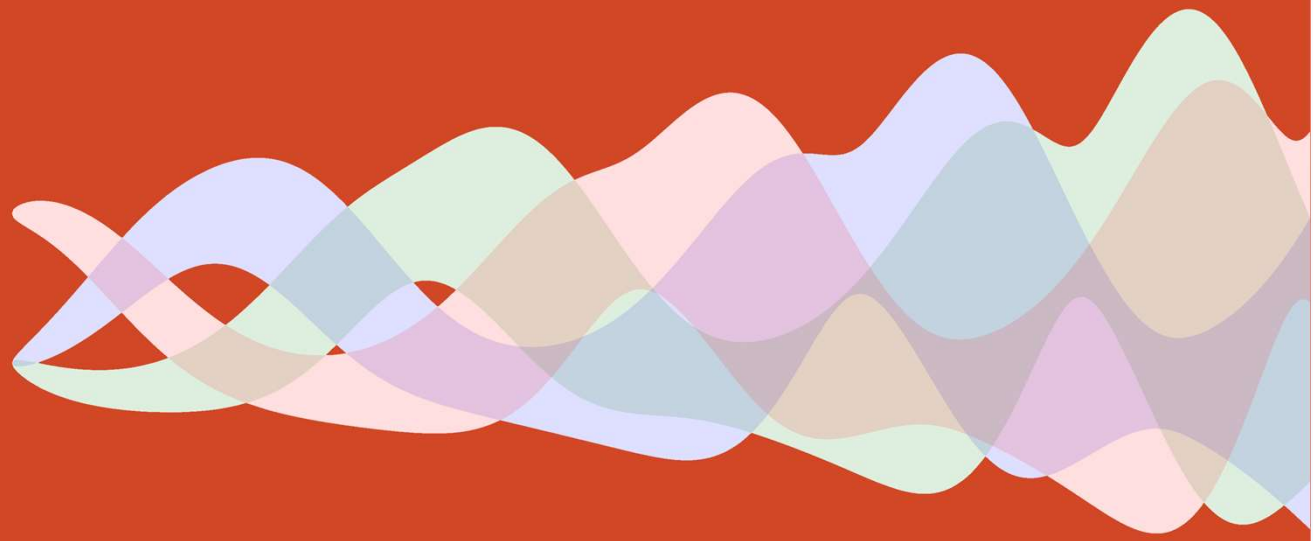
# Kaemika User Manual



# Installation

macOS https://apps.apple.com/us/app/kaemika/id1493299038?mt=12

iOS https://apps.apple.com/app/id1491803017

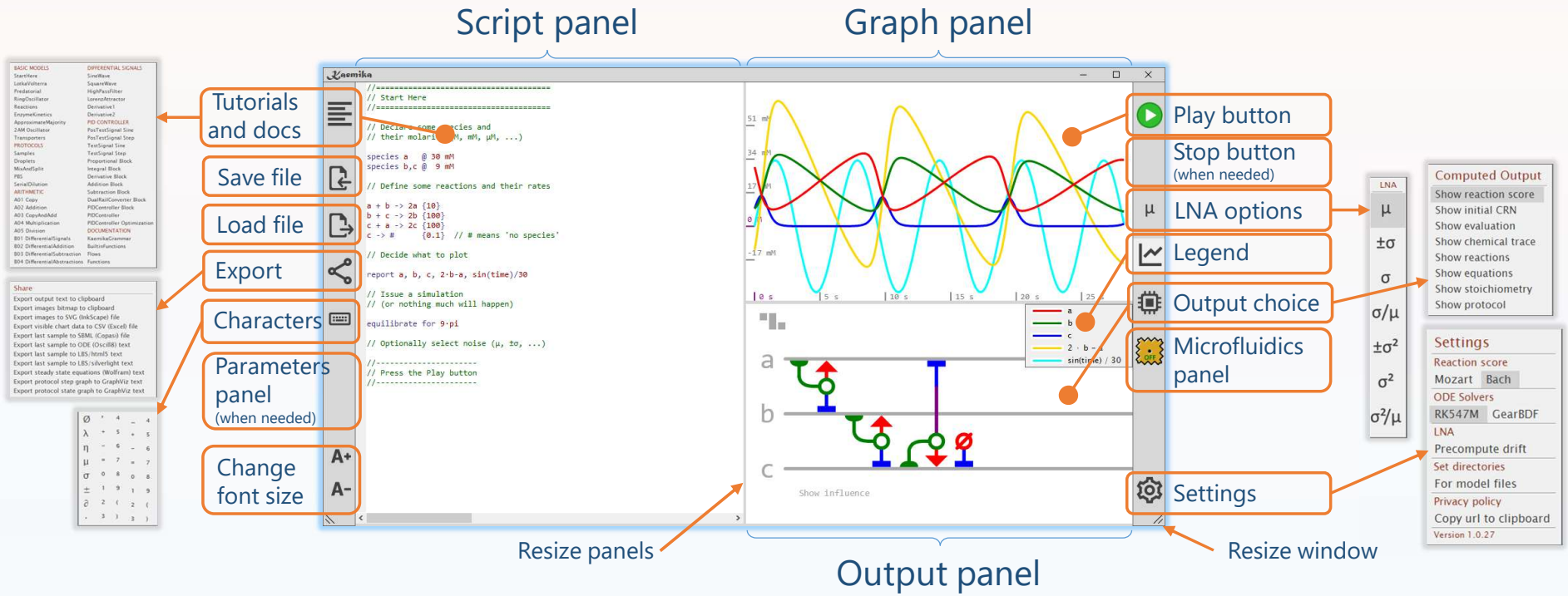
Android https://play.google.com/store/apps/details?id=com.kaemika.Kaemika&hl=en_GB

Windows https://www.microsoft.com/en-us/p/kaemika/9n258rnwv8pr?activetab=pivot:overviewtab

# Sources

GitHub https://github.com/luca-cardelli/KaemikaXM
(installation from GitHub is not supported)

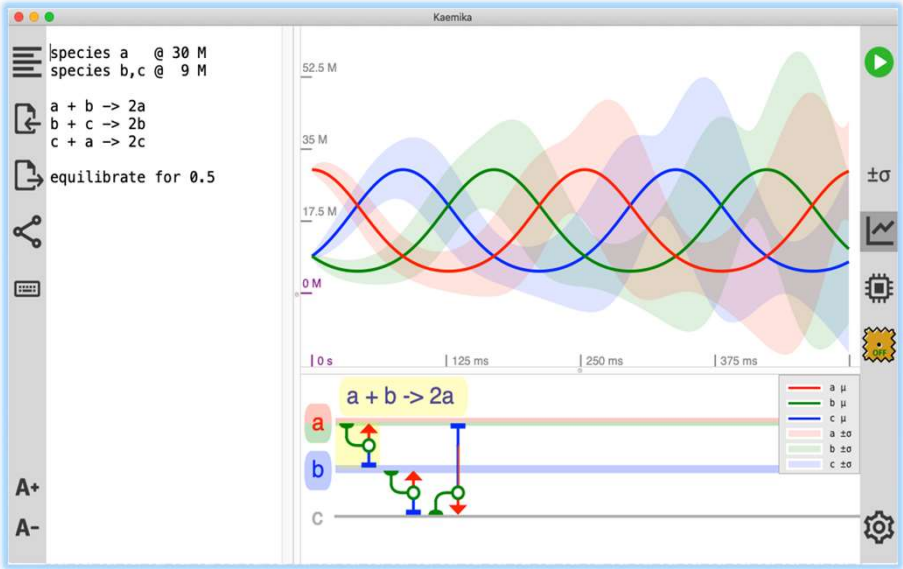
# Panels and Menus



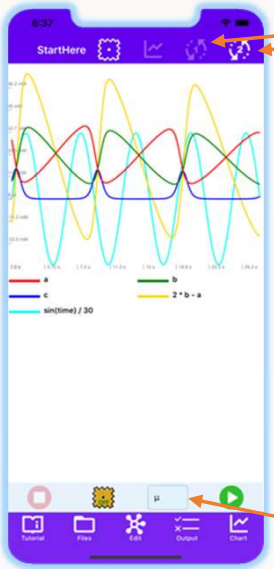

**Windows version**

Main interface view: showing the default script “Start Here” and its output after the app is first opened and the play button is pressed.

# Panels and Menus








Choice of
ODE solvers

**Graphical output is touch-enabled**
1-finger tap: select or activate
1-finger double-tap: activate
1-finger drag: move
2-finger tap: reset pan and zoom
2-finger drag: pan
2-finger pinch: zoom

LNA options

**macOS version**

Functionally identical to the Windows version.

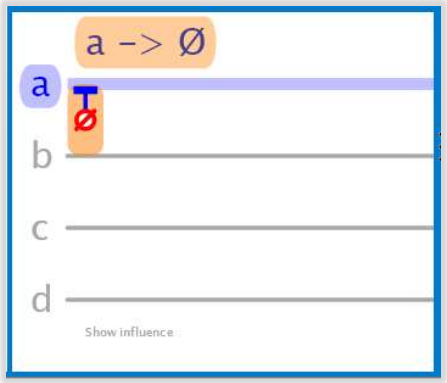
**Android/iOS versions**
Nearly identical functionality to Win/Mac, but arranged in 5 pages directly accessible from the bottom tabs:

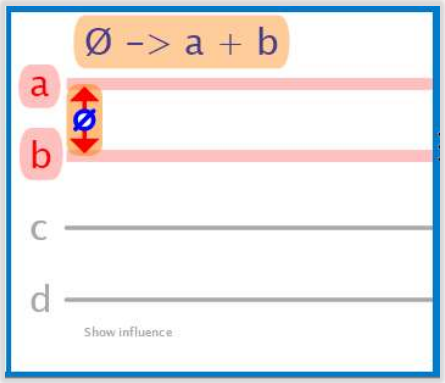
- Tutorial (built-in scripts and docs)
- Files (user scripts repository)
- Edit (script editor)
- Output (Graph&Text Output and Export Menu)
- Chart (Plots and Legends, and Microfluidics)

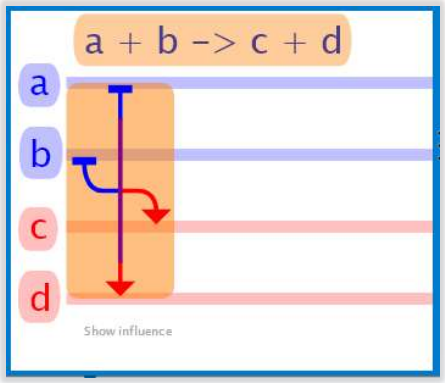
# Reaction score (graphical representation of reaction network) examples

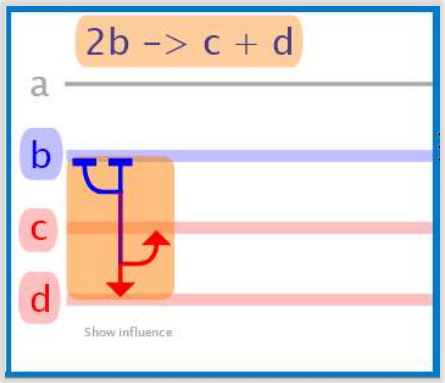
Horizonal lines: *species*. Vertical stripes: *reactions*.

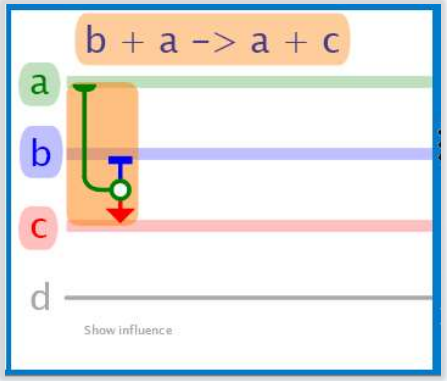
Blue: reagents. Red: products. Green: catalysts.

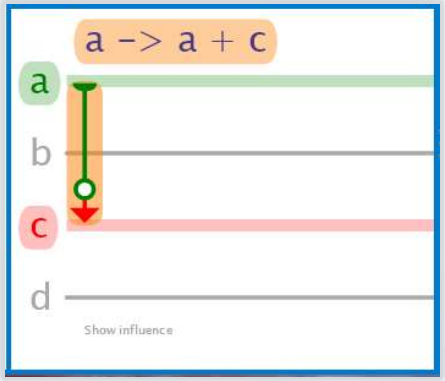


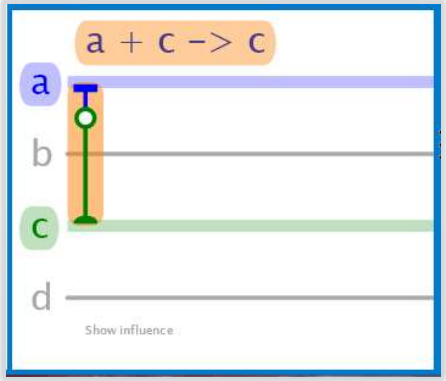



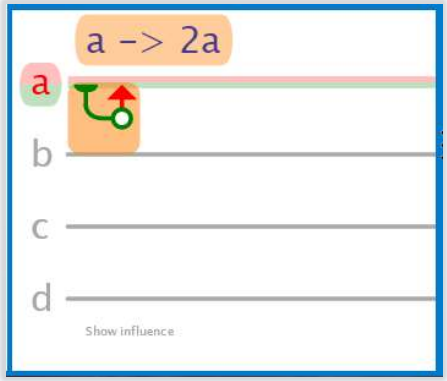



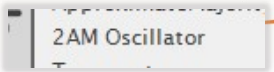



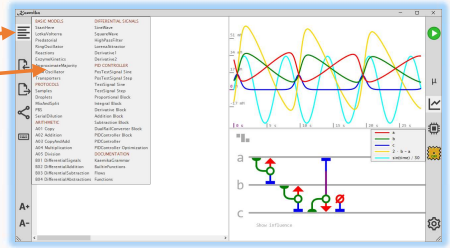



# Operation: Chemical reaction network simulation

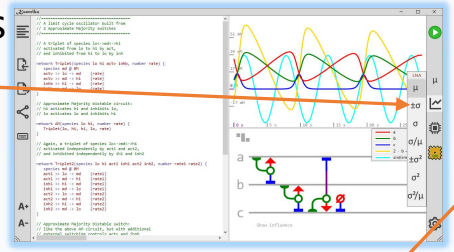
1. Select a script



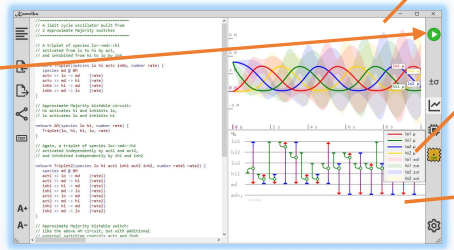



2. Select simulation options

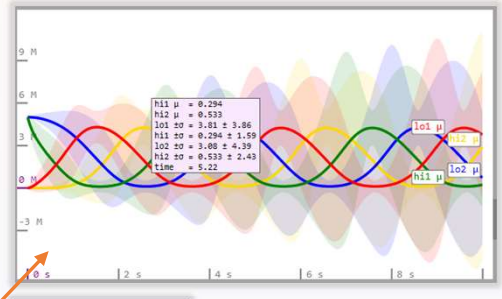
(stochastic LNA, ±σ)

3. Press the Play button



4. Results:



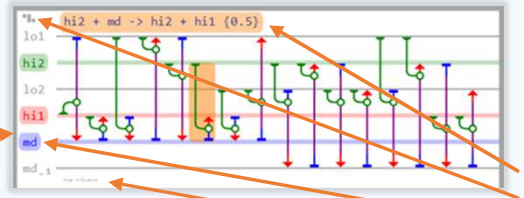

## Simulation plot

(mean ± standard deviation)

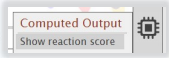
Hovering on trajectories reveals values.
Hovering on species tags reveals ODEs.
Pan&Zoom: mouse(Win/macOS) or touch(Android/iOS)


## Legend

Click to hide/unhide plot trajectories.
Shift-click to focus on a single trajectory.
Shift-click such a single trajectory to unfocus it.




## Reaction score



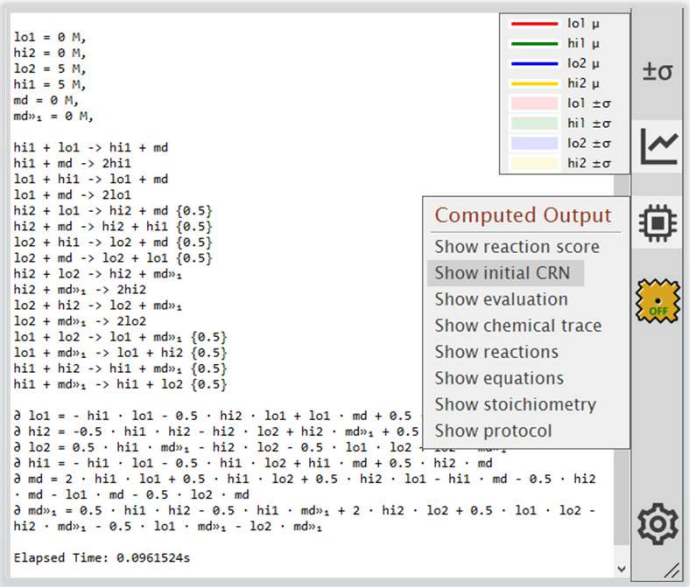

Graphical representation of the generated chemical reactions, as a directed multigraph

Hovering reveals reaction text.
Bin-packing.
Drag species tags to reorder them.
Highlight reaction relationships while hovering.

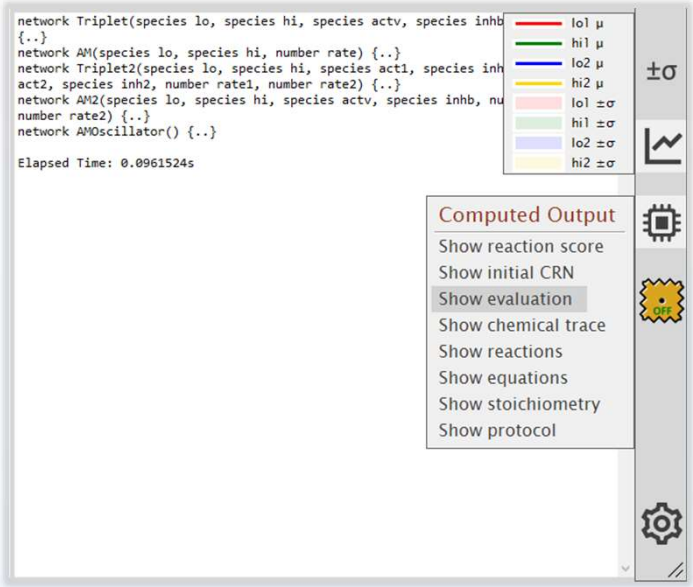
# Other outputs from the Computed Output menu

**Show initial CRN**
A self-contained summary of initial conditions, reactions, and ODEs generated by the input script.

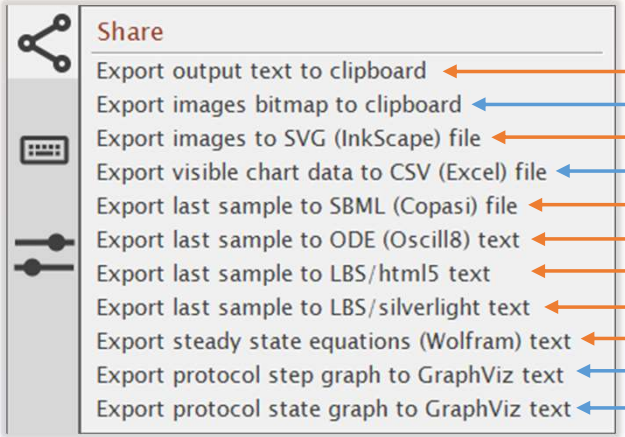
Further details are in
**Show reactions/equations/stoichiometry**
especially for multi-stage simulations (protocols).




**Show evaluation**
List of entities defined by input script evaluation, and any computational results.
**Show chemical trace**
Chemical events that occurred during evaluation.
**Show protocol**
Protocol events that occurred during evaluation.



# Export




Use the system clipboard to export (and import) any text.

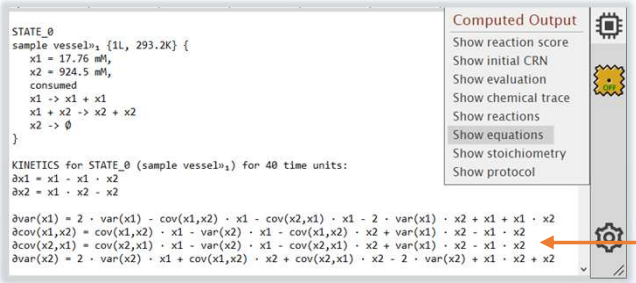
All graphics can be exported as bitmaps to the clipboard.

All graphics can be exported to file in resolution independent Scalable Vector Graphics format, for help with publication.

Simulation data, as filtered by the Legend, can be exported to file in Comma-Separated Values .csv format.

CRNs and ODEs can be exported to SBML and some other tools, and anyway they can be copied out as text and adapted as needed. Applies to the last simulated sample.

Protocol graphs can be exported in textual GraphViz format. (They are not graphically rendered in the Windows/MacOS version; they are graphically rendered in the iOS/Android version, and there they can be exported as text through the clipboard.)

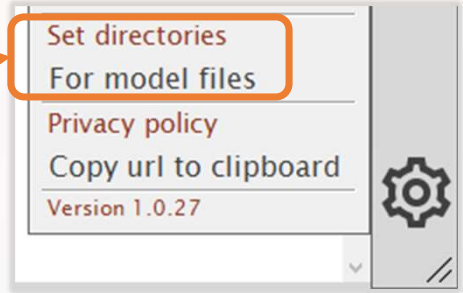



In addition, the “Show equations” option in the “Computed output” menu generates (when LNA is enabled) the symbolic LNA equations for covariances, which can be analyzed in more appropriate tools such as Mathematica.

# Saving, Loading, Editing scripts on Windows/MacOS

## Directory

Right away, use the *Settings* icon on the bottom right to set a convenient default directory for your scripts.



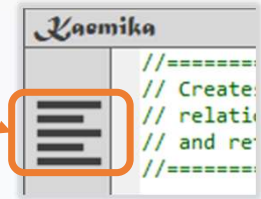

## Files

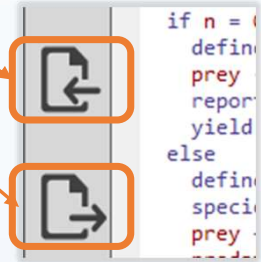
Start by selecting a script from the *Tutorial* menu. It's a copy, you can edit it.

Use the *Save* icon to save the current script through the standard system dialog.

Use the *Load* icon to load a script through the standard system dialog.

You can also cut and paste into the script window from the system clipboard.



## The script

The "current script" does not have a name; all saves are "Save As".

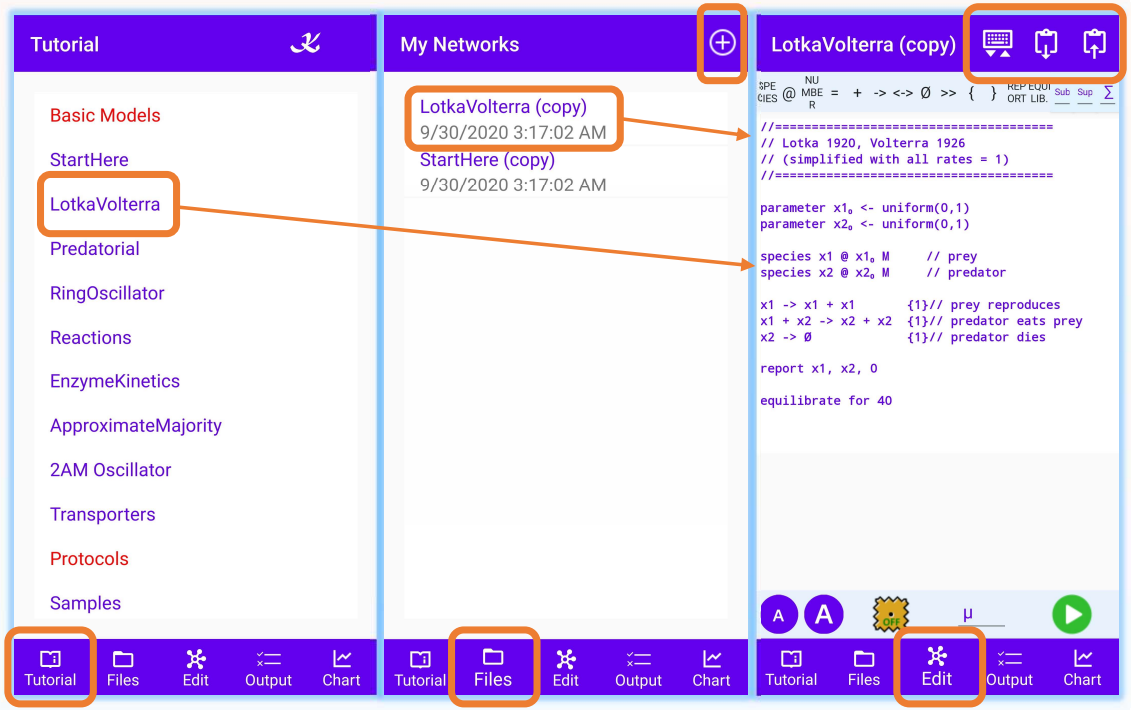
There is no list of saved scripts other than what you manage yourself in your directory.

Changes are saved automatically, in the sense that the last script will be reloaded if you close (or crash) and reopen the app.

## Backup

Set the default directory to your user space, to make sure that your scripts are not lost if the Kaemika app is updated or installed elsewhere.

# Saving, Loading, Editing scripts on Android/iOS




Tutorial

## Tutorial tab (built-in scripts)

Click an *item* in the list to open it and auto-switch to the *Edit* tab.
Then click the *Pencil* icon at the top to enable modifications.
Modified tutorial scripts are automatically added to the *Files* tab.

Files

## Files tab (user scripts, initially empty)

Click an *item* in the list to open it and auto-switch to the *Edit* tab.
***Android:*** press-hold an *item* in the list to rename or delete it.
***iOS:*** swipe left an *item* in the list to rename or delete it.
You can create a new script from the *Plus* button (top right).

Edit

## Edit tab (single-script editing)

If coming from *Tutorial*, click the *Pencil* icon at the top to enable editing.
An additional tool bar at the top contains some common input strigs.
Click the *Keyboard* icon to bring the system keyboard up or down, if needed.
The clipboard icons at the top can be used to *Copy* the whole script to the system clipboard, or to *Paste* (overwrite) the whole script from the system clipboard (repeat *Paste* to **undo**).

## Backup

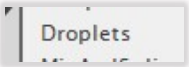
Through the system clipboard, you can backup a script to any other app or storage system: the Kaemika app has no direct access to locations outside of its domain.
WARNING: the *Files* tab scripts may get erased if you reinstall or update Kaemika.

## Undo text changes

***iOS:*** *shake the phone* to undo; also, on iOS 13+ *left-swipe with 3 fingers* while the keyboard is up.
***Android:*** the *Hacker's Keyboard* from the Play Store, which swaps in for the built-in keyboard, supports undo via Ctrl-Z, as well as Ctrl-X/C/V. There is *no* Android-wide way to undo text changes!

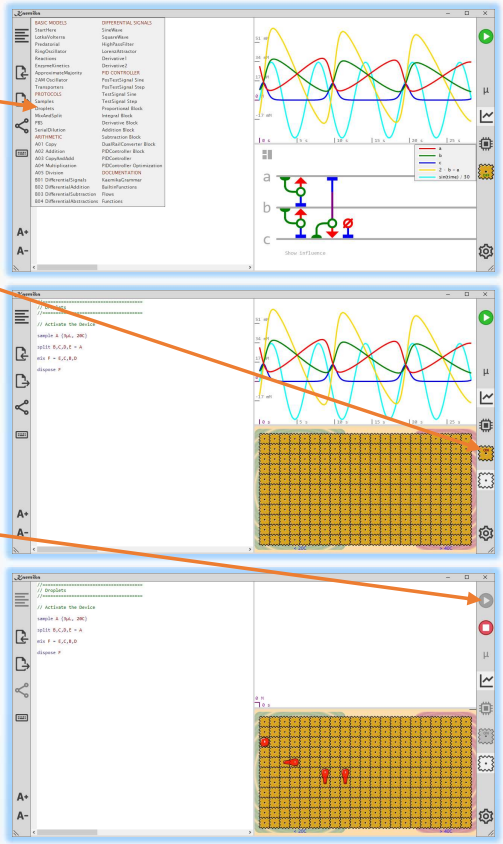
# Operation: Digital microfluidics

1. Select a script

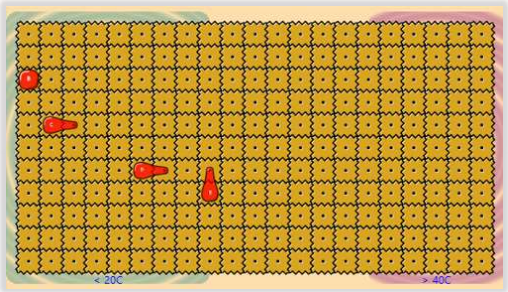



2. Activate microfluidics


3. Press the Play button



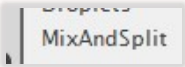
4. Results:



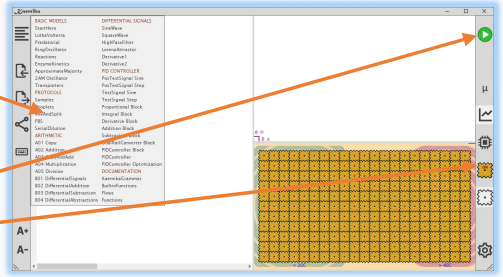
Droplets will split and merge according to the script, routing themselves to the right spots. But there is no chemistry in the Droplets script.

# Operation: Mixed simulation and microfluidics protocol

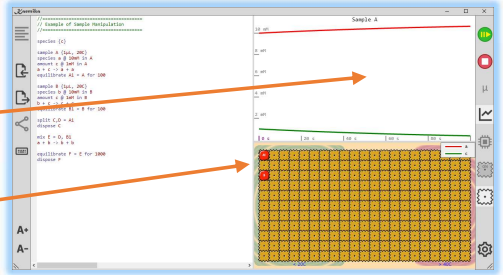
1. Select a script




2. Press the Play button (with microfluidics on)

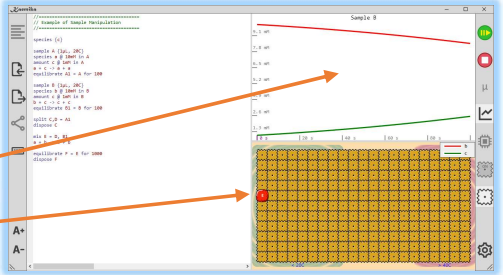


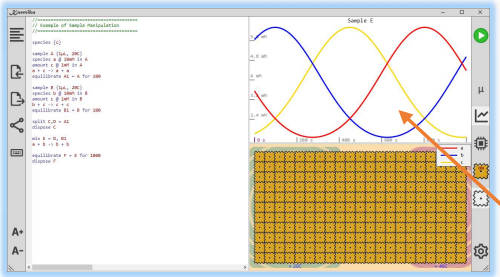
3. One droplets moves to a "warm" spot. It stops there, and a simulation runs. After that, it goes back home. A second droplet appears.



4. Press the (now paused) Play button

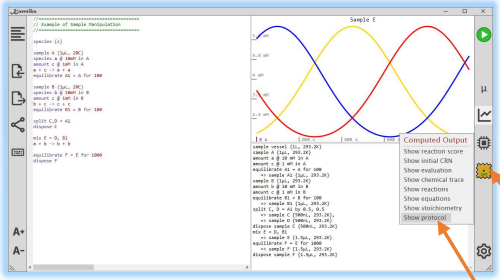
5. The second droplet moves, simulates, and then combines with the first one.



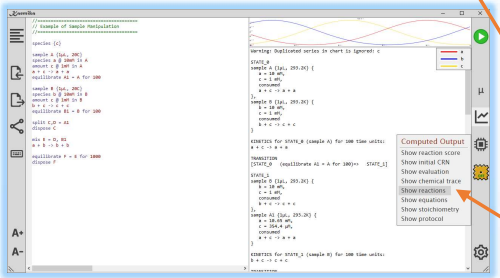


6. Press the (now paused) Play button

7. The combined droplet moves, simulates, and is recycled. This final simulation combines the reagents of the two initial droplets



8. Shift-click the Play button to do it again without pauses.

9. Turn off microfluidics. See a summary of the protocol steps in in "Show protocol", and of the chemical kinetics in "Show reactions"

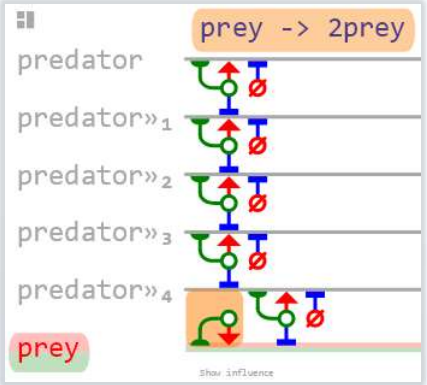
# Predatorial

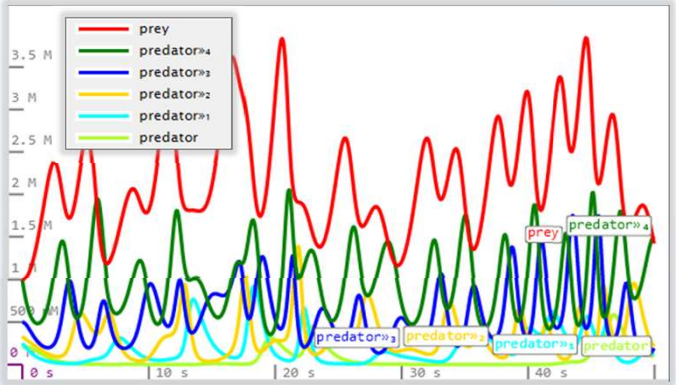
```
function Predatorial(number n) {
  if n = 0 then
    define species prey @ 1 M
    prey -> 2 prey
    report prey
    yield prey
  else
    define species predator @ 1/n M
    species prey = Predatorial(n-1)
    prey + predator ->{n} 2 predator
    predator -> Ø
    report predator
    yield predator
  end
}

species apexPredator = Predatorial(5)
equilibrate for 50
```

<- Make a stack of n predator-prey networks, each predator feeding on the next one
<- If n=0 there is only prey, no predators
<- Define and initialize the prey species
<- *Chemical reaction:* the prey reproduces
<- Report the prey for plotting
<- Return the prey species as the result of the function
<- Else if n>0
<- Define and initialize a predator species
<- Its prey is the result of Predatorial(n-1), the next species down the stack
<- *Chemical reaction:* predator eats prey and reproduces
<- *Chemical reaction:* predator dies (if it does not find prey quickly enough)
<- Report this predator species (there will be many) for plotting
<- Return the predator species as the result of the function

<- Make a stack of 5
<- Simulate



**Basic Scripting**

# Literals, Functions, and Operators for base types

**bool** true | false | not $b$ | $b_1$ and $b_2$ | $b_1$ or $b_2$ | $b_1 \overset{\text{equal}}{=} b_2$ | $b_1 \overset{\text{not equal}}{<>} b_2$

**number** 0 | 1 | -2.3 | 4.5e-67 | … | -$n$ | $n_1 + n_2$ | $n_1 - n_2$ | $n_1 \cdot n_2$ | $n_1 / n_2$ | $n_1 \wedge n_2$ | $n_1 \overset{\text{ascii}}{*} n_2 \equiv n_1 \overset{\text{unicode}}{\cdot} n_2$
$n_1 = n_2$ | $n_1 <> n_2$ | $n_1 > n_2$ | $n_1 < n_2$ | $n_1 >= n_2$ | $n_1 >= n_2$ | int($n$) (round to integer) | pos($n$) (round to non-negative real) |
abs($n$) | arccos($n$) | arcsin($n$) | arctan($n$) | arctan2($n_1$,$n_2$) | ceiling($n$) | cos($n$) | cosh($n$) | exp($n$) | floor($n$) |
log($n$) (base e) | max($n_1$,$n_2$) | min($n_1$,$n_2$) | sign($n$) | sin($n$) | sinh($n$) | sqrt($n$) | tan($n$) | tanh($n$) |
pi | e | maxNumber | minNumber | positiveInfinity | negativeInfinity | NaN

**string** “” | “abc” | “de\”fg\\hi” (escape) | … | $s_1 \overset{\text{concat}}{+} s_2$ | $s()$ (length) | $s(n)$ (0-index) | $s(n_1,n_2)$ (substring start,length) | basename($sp$) (non-α-converted name of species as string) | $s_1 = s_2$ | $s_1 <> s_2$

**list** [] | [1, 2, 3] | [[“a”,”b”],[“c”,”d]] | … | $l_1 \overset{\text{concat}}{++} l_2$ | $l()$ (length) | $l(n)$ (0-index) | $l(n_1,n_2)$ (sublist start,length) | $l_1 = l_2$ | $l_1 <> l_2$ |
map($fun$,$l$) | each($net$,$l$) | filter($fun$,$l$) | foldl($fun$,$z$,$l$) | foldr($fun$,$z$,$l$) | sort($fun$,$l$) | reverse($l$) | transpose($l$)

**species** a | b | c | … | $sp_1 = sp_2$ | $sp_1 <> sp_2$

**function** λ(){3} | λ(x){x} | λ(number n){n+1} | λ(function f){define bool b = f(0)>0 yield b} | … fun ≡ λ (ascii, unicode)

**network** η(){species s @ 3mM; s + s -> Ø} | η(species s){s -> Ø; Ø -> s + s} | … # ≡ Ø (ascii, unicode) net ≡ η (ascii, unicode)

**sample** sample A {1mL, 20C} (volume, temperature) | mix $S_0 = S_1,\ldots,S_k$ | split $S_1,\ldots,S_k = S_0$ by $n_1,\ldots,n_k$ (proportions) | dispose $S_1,\ldots,S_k$ |
regulate $S_1,\ldots,S_k$ to 25C (temperature) | concentrate $S_1,\ldots,S_k$ to 2mL (volume) | equilibrate $S_1,\ldots,S_k$ for 12 (seconds)

# Averaging simulation runs

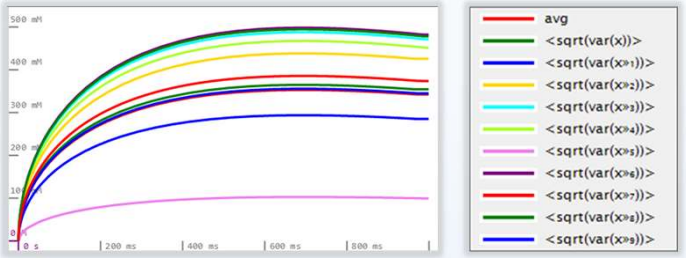
```
function run(number i) {
   define
      sample S
      number x0 = <-uniform(0,1)
      species x @ x0 M in S
      x -> Ø
      report y = sqrt(var(x)) in S
      equilibrate S for 1
   yield y
}
```

<- Make a function to run one simulation (i is an iteration index)
<- define *D* yield *E* returns the value of *E* after executing the statements *D*
<- Make a new sample S to contain species and reactions for simulation
<- Draw an initial value $x_0$ from a uniform distribution
<- Initialize a new species x to that value, and place it inside S
<- The reaction network for S (using just one reaction as an example)
<- report the s.d. sqrt(var(x)) of x into a *timeflow* y extracted from S
<- Simulate (and plot) sample S for 1 sec, with LNA enabled for var(x) to work
<- Return the *timeflow* y (i.e., the full trajectory of sqrt(var(x)))

```
list L = draw 10 from run

report foldl(fun(a b){a+b}, 0, L)/10 as "avg"
each(net(f){report f}, L)
equilibrate for 1
```

<- Invoke run(i) for i = 0..9, making a list L of 10 (randomized) timeflows
(Shift-click the Play button, or it will pause at every simulation!)
<- Fold* the average of the 10 timeflows from L into a new report, "avg"
<- report also each* of the 10 timeflows from earlier simulations
<- Run a final simulation to combine all the reports in a new plot




The 10 s.d. timeflows and their average can now be exported to file

*fun(..){..} is a nameless function, net(..){..} is a nameless network (a function with no value)
foldl and each are list iterators over functions and networks respectively

**Advanced Scripting**

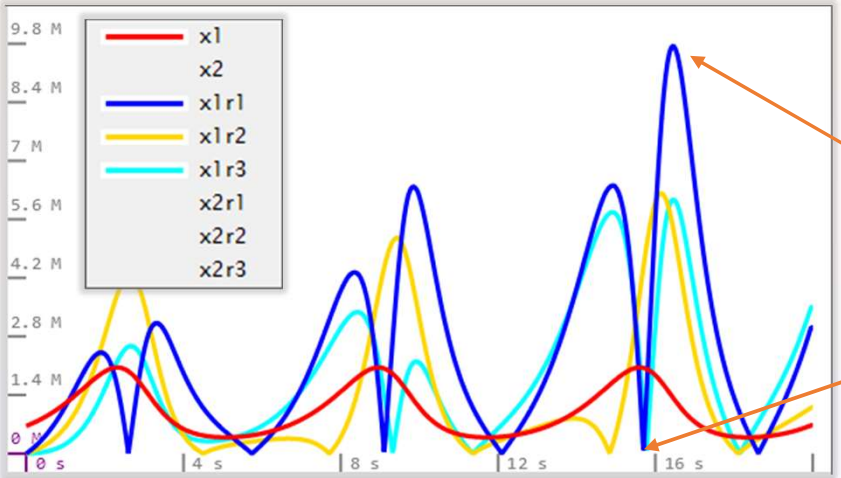
# Local Sensitivity Analysis (of a Lotka-Volterra system)

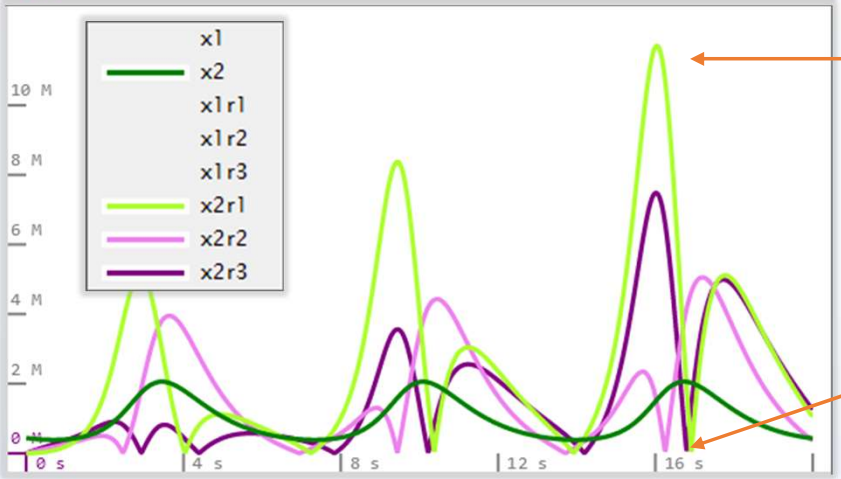
```
function f(number r1 r2 r3) {                    <- A function to run one simulation (ri are the input parameters to be perturbed)
  define                                         <- define D yield E returns the value of E after executing the statements D
    sample S                                     <- Make a new sample S to contain species and reactions for simulation
    species x1 @ 0.66 M in S                     <- Lotka-Volterra prey species (initial conditions could be a parameter as well)
    species x2 @ 0.44 M in S                     <- Lotka-Volterra predator species (initial conditions could be a parameter as well)
    x1 -> x1 + x1          {r1}                  <- Prey x1 reproduces, with rate r1
    x1 + x2 -> x2 + x2     {r2}                  <- Predator x2 eats prey, with rate r2
    x2 -> Ø                {r3}                  <- Predator dies, with rate r3
    report t1 = x1, t2 = x2 in S                 <- Report the timeflow (full trajectory) t1 for x1, and t2 for x2
    equilibrate S for 20                         <- Simulate the system: this will compute the timeflows t1,t2 (without plotting them)
  yield [t1, t2]                                 <- Return the output timeflows t1,t2 affected by the parameters r1,r2,r3
}

number d = 0.0001                                <- Perturbation value
[[t1, t2], [t1r1, t2r1], [t1r2, t2r2], [t1r3, t2r3]] =   <- Obtain a matrix of the 2 system outputs ti and their 3 individual perturbations tirj
  [f(1,1,1), f(1+d, 1, 1),f(1, 1+d, 1), f(1, 1, 1+d)]   <- (Shift-click the Play button, or it will pause at every simulation!)

report t1 as "x1", t2 as "x2",                   <- Prepare to report a plot of the sensitivities abs((ti – tirj)/d)
  abs((t1-t1r1)/d) as "x1r1", abs((t1-t1r2)/d) as "x1r2",   using the timeflows ti,tirj previously computed
  abs((t1-t1r3)/d) as "x1r3", abs((t2-t2r1)/d) as "x2r1",
  abs((t2-t2r2)/d) as "x2r2", abs((t2-t2r3)/d) as "x2r3"
equilibrate for 20                               <- Run a final simulation just to combine all the reports in a new plot
```

# … continued




The plots show the instantaneous sensitivities of x1,x2 with respect to the rate parameter r1,r2,r3 (separately), over the entire timecourse

x1 is usually more sensitive to r1 than to r2,r3

a point near the *peak* of x1 is *not* sensitive to r1 or r3 and instead is most sensitive to r2

x2 is usually more sensitive to r1 than to r2,r3 (and x2's peak sensitivity exceeds x1's)

a point near the *peak* of x2 is *not* sensitive to r1 or r3 and instead is most sensitive to r2

**Advanced Scripting**

# Global Sensitivity Analysis (of a Lotka-Volterra system)

```
function f(number r1 r2 r3) {
  define
    sample S
    species x1 @ 0.66 M in S
    species x2 @ 0.44 M in S
    x1 -> x1 + x1          {r1}
    x1 + x2 -> x2 + x2     {r2}
    x2 -> Ø                {r3}
    equilibrate S for 2.5
  yield [observe(x1,S), observe(x2,S)]
}
```

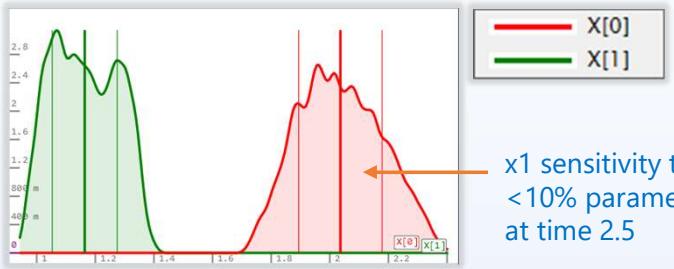
<- A function f to run one simulation (ri are the input parameters to be perturbed)
<- define *D* yield *E* returns the value of *E* after executing the statements *D*
<- Make a new sample S to contain species and reactions for simulation
<- Lotka-Volterra prey species x1 (initial conditions could be a parameter as well)
<- Lotka-Volterra predator species x2
<- Prey reproduces, with perturbed rate r1
<- Predator eats prey, with perturbed rate r2
<- Predator dies, with perturbed rate r3
<- Simulate the system up to time 2.5 (first peak of the oscillation)
<- Return the output concentrations of x1,x2 from S at time 2.5 as pairs

```
random X(omega w) {
   f(1+(w(0)-0.5)/10, 1+(w(1)-0.5)/10, 1+(w(2)-0.5)/10)
}
```

<- Create a bivariate random variable X over uniform[0..1) sample spaces w(i)
<- producing random instances $f(1+e_1, 1+e_2, 1+e_3) = [x1,x2]_{e1,e2,e3,t=2.5}$
with $e_1$, $e_2$, $e_3$ being 10% independent perturbations of the parameters

```
draw 2000 from X
```

<- Produce a density plot of 2000 instances drawn from X
i.e. a plot of the distributions of X[0]=x1 and X[1]=x2 at time 2.5
vertical bars are mean and standard deviation




x1 sensitivity to random <10% parameter variations at time 2.5

N.B., consider also exporting your Kaemika model to SBML and use the Sobol' method of global sensitivity analysis in e.g. Copasi.